Amendments to the Specification

Applicant respectfully requests that the Specification be amended by replacement paragraphs as follows:

[0035] Fig. 2 shows a travel itinerary form 200 that has been rendered similar to travel itinerary form 100 seen in Fig. 1, where some of the data-entry fields have been filled in. Here, the rendered form is generated after data was input by a user into the trip start date data-entry field 106 ~~206~~ as “03/13/2002”. For instance, an electronic forms application can produce a rendering file, shown as the data file tree 102, that is rendered to output the travel itinerary form 200 to a display. In this example, a trip start date data-entry field 204 corresponds to an event start date node 206 ~~202~~, and a trip end date data-entry field 208 ~~212~~ corresponds to an event end date node 210.

[0046] At block 314, the system receives the data that was input into the data-entry field. For example, the data entered into the data-entry field 412 by the user is received from the user through the user’s use of one or more input devices and the user interface. The system can receive the data character-by-character, when the data-entry field is full, or when the user attempts to continue, such as by tabbing to move to another data-entry field 414. In the foregoing example, the system receives “1/27/2002” from the user when the user attempts to advance to the next data-entry field.

[0068] The user can select the address block as whole and use the "Insert/Replace with" menu items or the user can right click with the user's mouse on the white space in the address block which selects the address block as well as brings up the context menu seen in Fig. 6b. Fig. 6b shows that the address block 606 is selected as a whole. The user now goes to the 'insert' menu and finds a fly out menu named "Replace with". The fly out contains, in this case, one entry titled "German Address" 604. If the user picks German Address 604, the current address block is removed and the new address block 704 is inserted as shown in Fig. 7b.

[0069] In Fig. 7a, the user has right clicked in the white space of the address block. The right click action by the user, in the context of the cursor being within the address block, will select the address block as a whole and surface the context menu. This context menu now contains the same "Replace with" fly out menu. If the user picks German Address 702, the current block is removed and the new block 704 is inserted as shown in Fig. 7b. The concept of the availability of replaceable blocks, and menu options defining the same with respect to the context of the cursor position, can be declaratively specified in a solution for a hierarchical data file into which data entered into the purchase request report electronic form will be stored.

[0096] Fig. 10 depicts an example 1000 of several files that make up or are referred to by an electronic form template 1020. The electronic form template

1020 can be a manifest of all files used by the electronic forms application 822 described herein. The electronic form template 1020 can be invoked when a user navigates to an XML document or when a new XML document is to be created. The electronic form template 1020 is a collection of files that declaratively defines the layout and functionality for an electronic form. Electronic form templates 1020 can be stored either as a single compressed file (e.g., with an "*.xsn" extension) or as a folder of files. This collection of files can be used in conjunction with any XML document to be opened and filled out with implementations of the electronic forms application 822. The electronic form template 1020 includes an XML schema definition 1006, a form definition 1004, and one or more XSLT filed 1008 for defining views.

[102] The runtime component 1104 includes an editor frame 1120 that includes XML editing 1122. The XML editing 1122 can function similarly to the electronic forms application 822. The editor frame 1120 bidirectionally communicates with a solution infrastructure 1124, such as XML solution 902 seen in Fig. 9. The solution infrastructure 1124 communicates with an XML store 1126 ~~1116~~. Each of the solution infrastructure 1124 and the XML store 1126 ~~1116~~ bidirectionally communicates with one of more XML documents 1130. Additionally, the solution infrastructure 1124 communicates with the one or more application files 1108. As seen in Fig. 9, the XML document 904 points to the solution definition 906 that should process the XML document 904 on the

computer 812. When the user uses the computer 812 to navigate to the XML document 904, the solution infrastructure 1124 loads the required the solution definition 906. If needed, the solution definition 906 handles any contextual user interfaces (UI), runs business logic associated with the XML document 904 (e.g., business logic 910, 412), controls data subscriptions for computer 812, creates local folders, searches and filters the local folders, and enforces security for all computer 812 operations. For some data operations, the XML solution infrastructure 1124 works with the local XML store 1126. The local XML store 1126 can provide electronic mail (e-mail) capabilities, such as an "inbox" and a "sent items folder" for XML payloads, and to enable the ordering, filtering and aggregation of XML data that is shredded or parsed in the local XML store 1126. The XML solution infrastructure 1124 allows a user of computer 812 to access various XML data sources on computer 812, in an intranet, as well as on an extranet or the World Wide Web. Given the foregoing, XML Documents 1130 can be displayed and edited using the XML Editing 1122 of the editor frame 1120.

[103] The solutions 1106 can be provided to a user of computer 812 as part of the architecture 1100, where the user would like to see samples or exemplary solutions 1140 from which the user can learn about the use and operation of electronic forms applications 1142 ~~822~~. Solutions 1106 can provide the user with a guide for customizing electronic forms and for building new solutions based on the exemplary solutions.

[104] Figs. 12a and 12b provide respective processes 1200A, 1200B by which a user of workstation 1202 can be provided with a solution having a corresponding electronic form that can be used by a user via an electronic forms application 1218. The electronic forms application 1218 and the workstation 1202 seen in Figs. 12a-12b can be similar to the electronic forms application 822 and the computer 812, respectively, as seen in Fig. 1. For instance, one of the form templates 1020 seen in Fig. 10 can be deployed to workstation 1202 so that the user of workstation 1202 can fill out the electronic form that corresponds to the form template 1020. As discussed with respect to Fig. 10, the form template 1020 includes the XML schema that is to be used, the formatting or presentation of the electronic form, and any logic that the electronic form uses. The deployment of the form template 1020 is available by process 1200A and process 1200B.

[105] In process 1200A, seen in Fig. 12a, the form template 1020 is deployed to a HTTP server 1210 (e.g., a Web Server). This deployment of the form template 1020 enables a transparent web deployment and maintenance model. Specifically, at block 1202 of Fig. 12A, a user opens a form of a certain type for the first time via an open request 1204 for an XML document 1206 using a URL 1208. The previously stored corresponding form template 1020 is deployed from HTTP server 1210 at a process flow 1214. The deployed form template 1020 is automatically downloaded <u>1212</u> on a network and stored as an "*.XSN" file <u>1216</u>

on the workstation 1202 being used by the user. The downloaded form template 1020 allows the user to use the form template 1020 even when the workstation 1202 is not connected to the network. Assuming the user has network connectivity, whenever the user opens a form, the electronic forms application 1218 can be configured to check to see if a newer version of the corresponding form template 1020 is available at a process flow 1214. If so, the newer version can be automatically downloaded to and stored on the users' workstation 1202 at a process flow 1214.

[106] Process 1200B, seen in Fig. 12b, is an alternative to process 1200A in that the form template 1020 can be deployed by a process flow 1224 directly to workstation 1202 from an information technology administrator 1226 in such a way that the form template 1020 will have access to local system resources and/or applications. In this case, the deployed form template 1020 can be packaged for execution via process flow 1224 (e.g., a "*.exe" or "*.msi" file) <u>or upload 1222 and storage as "*.XSN" file 1216</u>. As seen in Fig. 12b, the workstation 1202 navigates to an XML file 1206 and issues an open file request at a process flow 1204. A URN is returned to workstation 1202 at a process flow 1220. Here, for instance, the form template 1020 can access a directory service to obtain a users' role in an organization, where users are required to be at a certain management level to approve an electronic form, such as a travel itinerary, a purchase request or other document used in the ordinary course of business. Once obtained, the

electronic forms application 1218 can use this information to execute the appropriate business logic for the purchase request. The form template 1020 may be deployed, for instance, along with other client code as part of a larger client deployment scenario.